Real-Time Content-Aware Resizing of Video

A Thesis
Presented to
The Academic Faculty

by

Matthias Grundmann

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

College of Computing
Georgia Institute of Technology
December 2008

Real-Time Content-Aware Resizing of Video

Approved by:

Dr. Irfan Essa, Advisor
College of Computing
Georgia Institute of Technology

Dr. Greg Turk
College of Computing
Georgia Institute of Technology

Dr. Frank Dellaert
College of Computing
Georgia Institute of Technology

Date Approved: November 17, 2008

To my parents, who always supported me.

ACKNOWLEDGEMENTS

I want to thank everybody who supported me and contributed to this thesis, first and foremost my advisor Irfan Essa. You gave me the opportunity to come to America and study in this wonderful country. Thank you for all your support and input! Also, I am honored to be able to call Greg Turk and Frank Dellaert my committee members, thank you so much.

Vivek, I am deeply thankful for your help, technical conversations and suggestions. Many things would not have been possible without you! I had a great time working with you and I am looking forward to many more occasions to come.

I want to thank my former advisor from TUM, Nassir Navab. Thank you for introducing me to the wonderful field of Computer Vision and supporting me in going to Georgia Tech. I know the stay turned out longer than planned, but I would not be here without you. I also want to thank Franziska Meier for proof-reading this thesis and her helpful suggestions.

Thank you so much, Mum and Dad for all the years of unconditional love and support. You always took care of me and I felt always loved and appreciated. Thank you for everything you did for me.

Samantha - especially to you, I want to say thank you! You are always supportive, adventurous and never afraid of change. There is not better woman I could have wished for! You are the love of my life and I will always love you.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

In this thesis, we propose a new method for content-aware resizing of videos in real-time. Our approach consists of two steps. First, we compute a set of non-salient pixels in linear time which, when being removed or duplicated, do not alter the general appearance of the video. This is an extension of Avidan and Shamir's [3] greedy seam-carving approach to video. Second, we generate a new representation of the video, so called multi-view videos that allow us to resize the video in real-time, i.e. while being watched. This representation can be computed very efficiently, the complexity is linear in the number of frames and linear in the number of pixels in a video.

Our technique works on a broad variety of videos and is computationally inexpensive enough to be executed by a vast range of devices. We compare our technique to our own implementation of a current state-of-the-art approach and show several convincing results obtained by our technique.

CHAPTER I

INTRODUCTION

1.1 Motivation

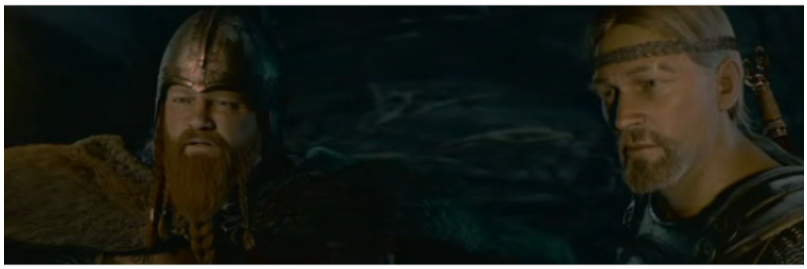
Videos come in various resolutions and have a broad range of various aspect ratios. While we can easily resize videos without altering the aspect ratio, resizing videos in only one of their spatial dimensions proves to be a challenge. For example, let us consider the case of resizing a 16:9 movie to 4:3 or nowadays the more likely case of resizing a 2.35:1 widescreen movie to 16:9. Established techniques used by the movie industry include pan and scan, which moves a cropping window of the desired target size to include the objects of interests or letterboxing where a uniformly scaled down version of the movie is matted onto the black background.

Although pan and scan uses all available pixels of the target device, content outside the cropping window is lost. Occasionally it is not even possible to cover all objects of interest by the cropping window as demonstrated in Fig. 1.1d. Letterboxing on the other hand does not discard any video content, however in order to preserve the aspect ratio some of device's pixels will not be used for displaying the video. These famous black bars on top and the bottom of the video can dissatisfy viewers especially if the device size is small or the change in aspect ratio large, e.g. 2.35:1 to 4:3.

Altering the aspect ratio of videos is not only of interest to the movie industry. Particularly nowadays, we use a range of different devices to watch videos. Smart phones, portable consoles and netbooks come in all different kind of resolutions and we wish to be able to use the whole screen of any small device to display the content of the video. Other applications are video sharing websites like YouTube^{TM} or MySpace^{TM} video which use a fixed resolution for video display but receive user-submitted content in different aspect ratios.

(a) Original frame.

(b) Frame resized to 74% of the original width.



(c) Frame resized to 126% of the original width.

(d) Pan and scan can not always preserve all important objects.

(e) Letterboxing does not use the whole possible display area.

Figure 1.1: Some examples of our surface carving technique. Comparison to established techniques like pan-and-scan and letterboxing are shown in (d) and (e). Original frame taken from the movie *Beowulf* - Copyright 2007 by Paramount Pictures.

1.2 Content-Aware Resizing

It is desirable to have a resizing technique that achieves both, preserving the content of the video while using all pixels of the desired target size. It is imperative that such a technique does not introduce spatial distortions like the ones we retain by non-uniform re-scaling. We can avoid these distortions and achieve both of above noted goals by content-aware resizing, i.e. preserving objects of interest while shrinking or enlarging unimportant features, like the background or homogenous regions. Therefore, a content-aware resizing technique does not simply scale the video along one axis, but squishes, enlarges or blends pixels and regions

(a) Original frame

(b) Frame resized to 74% of the original width

(c) Frame resized to 126% of the original width

Figure 1.2: Some examples of our surface carving technique. Original frame taken from the animated movie *WALL-E* - Copyright 2008 by Pixar Animation Studios.

with the aim to preserve the general appearance of the video. We will define what we mean by preserving the general appearance in chapter 2.

While some approaches have been proposed for content-aware resizing of videos [13, 6, 18], none of them has been demonstrated to work in real-time. Since the target's resolution is often not known beforehand, real-time performance is a crucial property for real-world applications. Our proposed algorithm is able to achieve real-time performance by efficiently computing a new representation for videos with a computational complexity that is linear in the number of frames and linear in the number of pixels within a video frame.

Known approaches for content-aware resizing can be separated in two classes, that we describe in the following sections.

1.3 Content-Aware Resizing posed as a global optimization problem

Wolf et al. [18] and Gal et al. [14] have suggested to pose content-aware resizing as a global optimization problem. The task is to compute a content-specific warping that resizes the

image or video to its desired target size. Usually, various constraints are imposed on this warping, e.g. specific features like faces or highly structured areas should be preserved or the warping for manually selected objects of interest is constrained to be a similarity [14]. Additional smoothing constraints are imposed, so that spatial or temporal artifacts are avoided.

Posing the problem as one global optimization problem comes at a cost though: For each target size a new warping needs to be computed. Because we have to determine the position of N pixels in the final image or video (here N refers to the total amount of pixels in an image), solving the optimization problem is usually computational quite expensive, even when formulated as a sparse least square problem ([18]). In consequence, the main disadvantage of this technique is that it can not be used for real-time resizing of images or videos, especially when we are focusing on mobile devices or average computers.

1.4 Iterative Content-Aware Resizing by employing a greedy approach

Avidan and Shamir [3] proposed a greedy content-aware resizing technique for images. Instead of computing an optimal warping, sets of pixels, so called seams, are computed that when being removed or duplicated alter the width or the height of the image by exactly one. When successively employed, this technique can resize the image to any desired (rectangular) destination size. An important property of this approach is that the successively computed sets of pixels are disjoint. That allows us to built a representation that specifies for every pixel the width or height when it was removed. Consequently, resizing a video in real-time is simply a comparison of the representation to the desired target width or height.

Note that a similar approach of using pre-computation can not be employed to the global optimization approach described above. For each target size we would have to save the computed warping, resulting in data that is magnitudes larger than the original video.

A possible disadvantage of greedy approach is, that the results might be not as good as the ones achieved by computing an optimal warping. A warping approach can achieve sub-pixel accuracy while computing disjoint sets of pixels constrains us to integer pixel

locations. Furthermore, a global optimization approach considers a larger set of low energy pixels. This can avoid some artifacts that might be introduced by the greedy approach. However, we will see in the experimental section in chapter 4 that the global optimization approach suffers from other artifacts that are difficult to control.

1.5 Other approaches

Other possible content-aware solutions are to crop the image or video to important regions, which is similar to pan and scan. Liu and Gleicher [13] propose the use of a salience map to crop the video or to compute a virtual pan of the camera. The saliency map is composed of image, motion and object salience for each frame. A similar saliency map is used by Fan et al. [8] to zoom into the region of most interest. Additionally, the user can shift the virtual camera to different regions of interest while watching the video.

Another approach is taken by Tao et al. [6]. They employ foreground extraction to each frame and create a density map from the foreground pixels. Clustering is used to compute a window that contains the highest density and belief propagation is employed to stabilize this window over time. Cropping to the computed window is the result of their resizing process. Because the orientation of the window is allowed to change between frames, unnatural camera movements may be introduced.

Setlur et al. [16] resizes images by separating the background from the foreground, where the foreground consists of objects of interest. The background is scaled to the desired target size and any holes are filled by in-painting. The foreground objects are uniformly resized and pasted on the background. This work could be extended to videos especially when combined with Wexler et al.'s [19] work to cover the space time holes. Wexler et al. use a global optimization technique to cover these holes in a video with patches that are extracted from the remaining video, which is very similar to Kwatra et al.'s [10] idea of synthesizing textures.

Recently, Rubinstein et al. [15] proposed the use of graph-cuts to compute smooth surfaces to resize video w.r.t. its content. Their approach is close to ours, however we use a dynamic programming approach to compute an approximate surface instead. While their

approach is guaranteed to find the optimal surface, their results show similar temporal coherence artifacts in case of fast moving objects. Note, that we could partially overcome this artifacts by our cascading surface approach. One advantage of our technique is its computational inexpensiveness compared to graph-cuts.

1.6 Outline of this thesis

We propose a fast algorithm for content-aware resizing of videos that extends Avidan and Shamir [3] greedy seam-carving approach to video. We demonstrate that it is capable of resizing a video in real-time, i.e. while we are watching it. To achieve real-time performance we compute an intermediate representation of the video, so called multi-view videos. This representation can be computed very efficiently, the complexity is linear in the number of frames and linear in the number of pixels in a video.

In chapter 2 we describe our approach for content-aware resizing of videos. We will build our algorithm on computing smooth surfaces which will enable us to alter the width of a video by exactly one while minimizing spatial and temporal artifacts. We will show that in case of fast motions a smooth surface is not always desirable and propose a solution that breaks surfaces along the temporal dimension while reducing the number of discontinuities per frame.

In order to compare to non-greedy approaches, we re-implemented Wolf et al.'s [18] optimization method. We derive the involved linear system explicitly in chapter 3 and show how it can be solved efficiently by solving for the normal equation. We also propose several additions to Wolf et al.'s [18] work.

Lastly, we will evaluate our approach on various videos in chapter 4 and compare our work to Wolf et al.'s [18]. We will show that our algorithm achieves similar or better results while having the advantage to work in real-ime.

CHAPTER II

OUR APPROACH

2.1 Seam-Carving revisited

Before we explain our approach in detail we will quickly describe Avidan and Shamir's [3] approach to resize images w.r.t. their content. Resizing w.r.t. image content means, that salient pixels or regions will be preserved in their size while non-salient regions or pixels are allowed to be squished together or spread out depending on whether one wants to shrink or enlarge an image. The notion of saliency is defined on a local per-pixel level as the sum of the absolute values of the image gradients. Specifically, the saliency S of a pixel p in an image I is simply defined as

$$S(p) = |\partial_x I(p)| + |\partial_y I(p)|. \tag{2.1}$$

The intuition behind this definition of saliency is that gradients of small magnitude usually hint at low-frequency, textureless regions. We expect that we can alter a textureless region without introducing too many visible artifacts. On the other hand, gradients of large magnitude hint at high-frequency, textured or noisy regions, i.e. details we want to preserve. Obviously this purely local definition has two shortcomings:

- By relying on image gradients, the definition of saliency in Eq. 2.1 is sensitive to noise. Furthermore, image regions that originate from natural scenes like water, grass or rocks have usually high frequency gradients due to their probabilistic nature. On the other hand, these regions make usually good candidates for resizing because due to their inherent noisy characteristics they hide seams very well, as observed by Agarwala et al. [1] and Kwatra et al. [11].

- While textureless regions might be unimportant, this is not always the case. We will show cases in our experiments in chapter 4 where the background is richly textured

(a) Vertical seam example

(b) Another vertical seam example

Figure 2.1: Vertical seams of minimal saliency.

while persons in the foreground wear textureless clothes. As a result, the persons are squished together which produces unpleasant results.

Having saliency defined, we can select a set of pixels of minimal saliency and remove or duplicate them in order to alter the size of an image. Avidan and Shamir's [3] note that this set of pixels does not only have to be minimal in saliency but also adhere a specific spatial coherence, otherwise artifacts like in Fig. 2.3b are introduced. In case we want to change one dimension of an image by exactly one pixel, spatial coherence can be achieved be requiring the set of pixels to form a *seam*. We distinguish horizontal and vertical seams depending on whether we want to alter the height or the width of an image by one. A vertical seam V is defined as a set of neighboring pixels that are connected, with only one pixel per row:

$$V = \{(x_i, i) : i = 1 \dots h, |x_i - x_{i-1}| \leq 1\} \tag{2.2}$$

where h denotes the height of the image. Similar a horizontal seam is defined as a set of neighboring pixels that are connected, with only one pixel per column. An example of two seams of minimal saliency is shown in Fig. 2.1.

Avidan and Shamir's [3] resize an image by iteratively removing or duplicating seams of minimal saliency. These seams are found by employing a dynamic programming approach,

similar to the one that we use to resize videos as described in the next section.

2.2 Overview of our system

Given a video sequence, our goal is to compute a set of pixels that when being removed or duplicated alter the width or the height of the video by exactly one while preserving the general appearance of the video. We will focus our description on decreasing the width by one. In section 2.7 we explain the opposite case of increasing the width by one. The case of altering the height is similar and can be achieved by transposing each frame and applying the same technique.

Let $\mathbf{V} \in \mathbb{R}^{m \times n \times r}$ be a video of dimensions $n \times m$ consisting of r frames with n being the width and m the height of the video. We will use matrix notation and denote the corresponding dimensions with y, x and t. In order to decrease the width by one, we have to delete a pixel from every of the m rows in each of the r frames. Consequently, the set of pixels being removed can be represented by a $m \times r$ matrix, where the value of each matrix element is an positive integer representing the x-value of the pixel. Let us denote this matrix by $\mathbf{S} \in \mathbb{N}^{(1..m)\times(1..r)}$. In order to maintain the general appearance of the video, the set of pixels represented by $\mathbf{S}$ should have two properties:

1. When being removed, $\mathbf{S}$ should not introduce visible artifacts. One way of achieving this is to require that $\mathbf{S}$ is a smooth surface along the spatial and temporal dimension of the video. Violation of spatial continuity distorts each frame, while violation of temporal continuity produces flickering videos. The two distortions are illustrated in Fig. 2.3. Because $\mathbf{S}$ holds positive integers, this smoothness constraint can be expressed in discrete terms by requiring that every pair of neighbors in a row or column of $\mathbf{S}$ differs by maximal one. Formally, we want:

$$\begin{aligned} \mathbf{S} = (s_{i,j})_{i,j} \qquad & \text{with } |s_{i,j} - s_{i+1,j}| \leq 1 \ \forall j \in \{1 \dots r\} \wedge i \in \{1 \dots m-1\} \\ & \text{and } |s_{i,j} - s_{i,j+1}| \leq 1 \ \forall i \in \{1 \dots m\} \wedge j \in \{1 \dots r-1\}. \end{aligned} \tag{2.3}$$

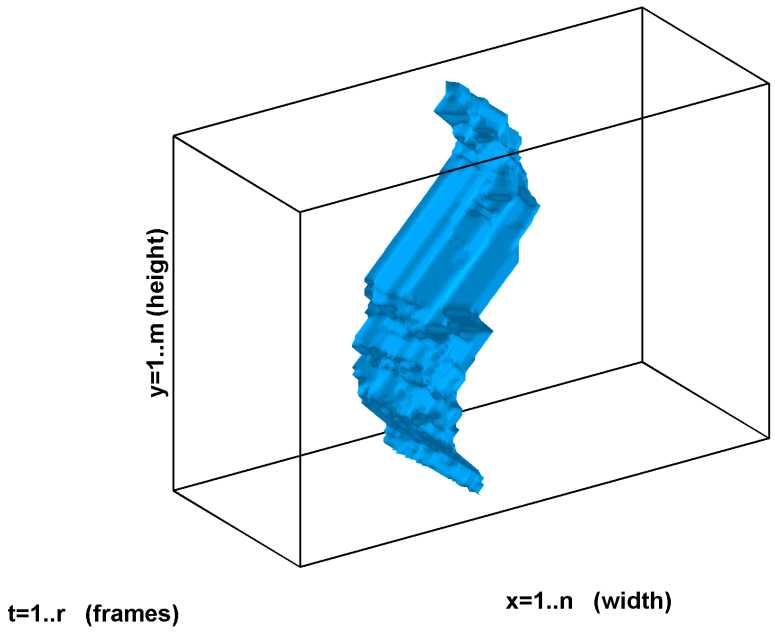


Figure 2.2: Example of a smooth surface in the video volume $\mathbf{V} \in \mathbb{R}^{m \times n \times r}$.

Thus, $\mathbf{S}$ represents a vertical seam (given by the restriction $\mathbf{S}|_t$) in each of the r frames, as well as m seams in the $x - t$ plane. Fig. 2.2 shows an example of $\mathbf{S}$.

2. The surface $\mathbf{S}$ should preferably cut through pixels that are non-salient. Pixels within homogeneous, low-textured regions make a perfect candidate because we can extend or shrink these regions without introducing noticeable artifacts. On the contrary, cutting through high-frequency objects like edges, straight lines or even faces is very noticeable. Homogeneous regions have the property that the magnitude of their image gradient is very small. Therefore, we define our cost measure similar to Avidan and Shamir's [3] saliency measure described in section 2.1 as

$$\mathbf{C} = |\partial_x \mathbf{V}| + |\partial_y \mathbf{V}| + |\partial_t \mathbf{V}| \tag{2.4}$$

with $\mathbf{C}$ having the same dimensions as $\mathbf{V} \in \mathbb{R}^{m \times n \times r}$. Consequently, we define the cost of the surface $\mathbf{S}$ to be the summed cost of its elements

$$C(\mathbf{S}) = \sum_{i=1..m, j=1..r} \mathbf{C}(S_{i,j}, i, j). \tag{2.5}$$

Remember that $S_{i,j}$ represents the x-location of the surface in the i-th column and the j-th frame. Furthermore, $C(\mathbf{i}, \mathbf{j}, k)$ denotes the matrix element $C_{\mathbf{j},\mathbf{i},k}$. Please note that we swapped the indices i and j in the definition of $C(i, j, k)$ to be able to use points as indices for the matrix. We will use this notation throughout this thesis.

Lastly, we are interested in finding the optimal surface $\mathbf{S}^*$ that has the minimal cost:

$$\mathbf{S}^* = \min_{\mathbf{S} \in \mathbb{N}^{(1..m)\times(1..r)}} C(\mathbf{S}) \qquad \text{with } \mathbf{S} \text{ satisfies Eq. 2.3.} \tag{2.6}$$

Different cost functions can be used with this framework in order to preserve faces, fast motions, etc. We discuss these cost functions in section 2.7.

Our system consists of three principal steps. First, we use a dynamic programming approach to quickly compute an approximate surface $\mathbf{S}$ that satisfies Eq. 2.6 for most of its elements. Our approach is approximate in the sense that some of its elements violate the smoothness constraint of Eq. 2.3. Second, we use an iterative weighted technique to smooth the surface w.r.t. the cost function. Generally, less than 50 iterations are sufficient. Third and finally, we compute and remove the computed surfaces iteratively and built a multi-view representation of the video that allows us to resize videos in real-time. We will discuss each step in the next sections.

2.3 Computing approximate surfaces

Computing the optimal surface $\mathbf{S}^*$ that satisfies Eq. 2.3 is a computational expensive task. Avidan and Shamir [3] solved the problem efficiently in an image by using a dynamic programming approach. Let us consider the case of a vertical seam. We know that the seam has to start at one pixel in the first row of an image and will end at another pixel in the last row. Furthermore it has only one pixel per row and satisfies the smoothness constraint in Eq. 2.2.

(a) Original frame

(b) Spatial continuity violated

(c) Resized frame k

(d) Temporal continuity violated (frame k+1). Scaled to 160%. for better visualization

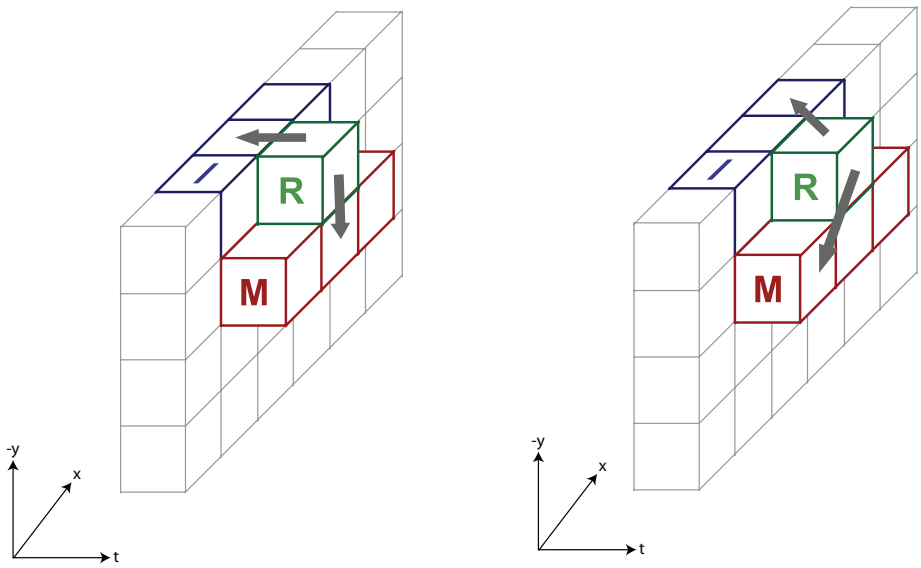
Figure 2.3: (a) and (c) Original frame and properly resized frame. (b) Lack of spatial continuity distorts resulting frame significantly. (d) The edge image of the previous frame is overlayed in red. Note, that the tracks, the wheelbarrow and the pillar in the bottom right move slightly to the right in the current frame. This introduces very noticeable jitter. See supplemental material for video. Original frame taken from the movie *There Will Be Blood* - Copyright 2007 by Paramount Vantage.

Applying dynamic programming in this setting means to express the total cost $\mathbf{M}$ of the minimum seam that ends in pixel $p = (x, y)$ recursively as a function of the pixels cost $\mathbf{C}(x, y)$ and the total cost of the minimum seam that ends in the row above $(y - 1)$:

$$\mathbf{M}(p) = \mathbf{M}(x, y) = \mathbf{C}(x, y) + \min_{i \in \{-1,0,1\}} \mathbf{M}(x + i, y - 1). \tag{2.7}$$

The minimal vertical seam is incident to the pixel $(\hat{x}, h)$ in the last row h that has the minimal cumulative cost $\mathbf{M}(\hat{x}, h) = \min_x \mathbf{M}(x, h)$. The x-coordinates of the seam can be determined by backtracking $\mathbf{M}$. A visualization example of $\mathbf{M}$ can be seen in Fig. 2.5a.

In case of video the vertical seam translates to a vertical surface. The surface has to



(a) Smoothness constraint fulfilled (b) Smoothness constraint violated

Figure 2.4: For each reference point $\mathbf{R}$ we first select the best neighbor $\hat{\mathbf{R}}$ in the row above (here row below for better illustration) based on the cumulative seam cost $\mathbf{M}$. Then we select the best neighbor $\bar{\mathbf{R}}$ in the previous frame, so that $|\mathbf{I}(\hat{\mathbf{R}}) - \bar{R}_x| \leq 1$, i.e. the smoothness constraint is not violated. $\mathbf{I}(\hat{\mathbf{R}}) := R_x$ for this illustration.

start at one seam at the $r \times n$ dimensional top of the video volume $\mathbf{V}$ and ends at a seam at the bottom of $\mathbf{V}$, as shown in Fig. 2.2. There are $O(n \cdot 3^r)$ possible seams for the top and bottom cap of the volume, leading to exponential complexity.

Instead, we propose a method that returns only an approximate surface, but in linear $O(mnr)$ time. The key idea is to iterate over each of the r frames and compute all possible vertical seams for each frame. For all frames except the first ($t > 0$) we also consider the costs of the best vertical seam in the previous frame that is adjacent to the current one. This previous seam is saved in a 3-dimensional matrix $\mathbf{I}$ for later use. Therefore each pixel $\mathbf{p} = (x, y, t)$ is assigned the cumulative minimum cost $\mathbf{M}$ of all possible smooth surfaces $\mathbf{S}$ starting at the first frame and ending at frame t at location (x, y).

Let us denote by $\hat{\mathbf{p}}$ the neighbor of minimum cumulative cost in the row above:

$$\hat{\mathbf{p}} = \underset{\mathbf{q}=(x+i,y-1,t)}{\operatorname{argmin}} \mathbf{M}(\mathbf{q}) \quad \text{with } i \in \{-1, 0, 1\}. \tag{2.8}$$

$\mathbf{I}(\hat{\mathbf{p}}) = \mathbf{I}(\hat{p}_x, \hat{p}_y, \hat{p}_t)$ specifies the x-offset of the best seam wrt. $\hat{\mathbf{p}}$ in frame $t-1$ and in row

$y-1$. Therefore, we can select the minimum element $\bar{\mathbf{p}}$ in row y in frame $t-1$ that meets the smoothness constraint of Eq. 2.3

$$\bar{\mathbf{p}} = \underset{\mathbf{q}=(x+i,y,t-1)}{\operatorname{argmin}} \mathbf{M}(\mathbf{q}) \quad \text{with } i \in \{-1,0,1\} \wedge |x+i-\mathbf{I}(\hat{\mathbf{p}})| \leq 1 \tag{2.9}$$

and define

$$\mathbf{I}(\mathbf{p}) := \bar{\mathbf{p}}_x. \tag{2.10}$$

In summary, for each pixel $\mathbf{p}$ we select its neighbor of minimal cost in the row above $\hat{\mathbf{p}}$ and its minimal neighbor in the frame before $\bar{\mathbf{p}}$. While we can determine $\hat{\mathbf{p}}$ from $\mathbf{M}$ via backtracking, we have to explicitly remember $\bar{\mathbf{p}}$ by saving its x-coordinate to $\mathbf{I}$.

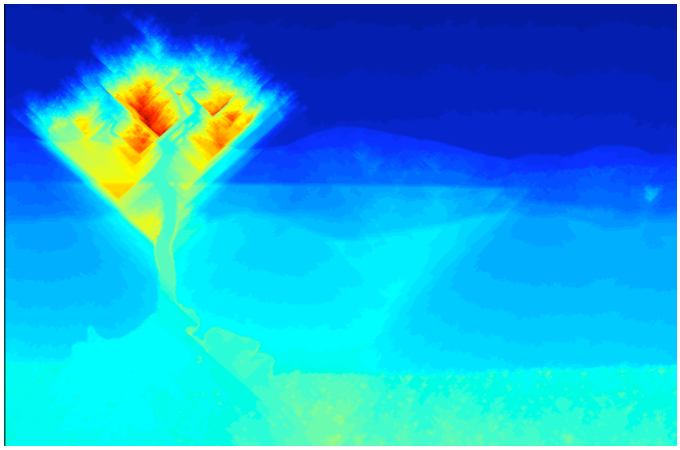
Similar to the image case in Eq. 2.7, the cumulative minimum cost of a seam incident to $\mathbf{p}$ is defined as the sum of the cost of $\mathbf{p}$ and the cumulative costs of the above described neighbors:

$$\mathbf{M}(\mathbf{p}) := \mathbf{C}(\mathbf{p}) + \mathbf{M}(\bar{\mathbf{p}}) + \mathbf{M}(\hat{\mathbf{p}}). \tag{2.11}$$

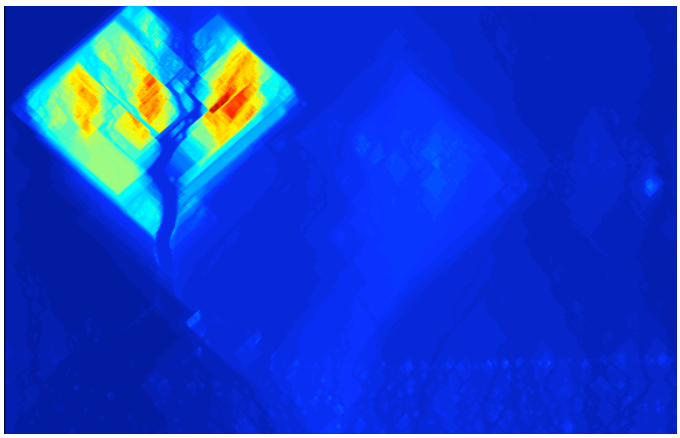
Fig. 2.4 illustrates this approach.

The design described so far has two shortcomings:

1. To properly assign the value of $\mathbf{I}(p_x, p_y, p_t)$ we want that the cumulative minimum cost of its three neighbors in the previous frame, $\mathbf{M}(p_x+i, p_y, p_t-1),\ i \in \{-1,0,1\}$ specifies the cost of the *complete* vertical seam plus those in the frames before. So far, $\mathbf{M}(p_x+i, p_y, p_t-1),\ i \in \{-1,0,1\}$ only specifies the cost of the *partial* vertical seam that ends in row p_y. Consequently the $\mathbf{M}$ values for $p_y = 1$ are much lower than those for $p_y = h$. To replace the current partial $\mathbf{M}$ values with the total cost of the seam we adopt a technique described by Ahmad and Choi [2]. After computing the $\mathbf{M}$ values in the described top-to-bottom row fashion for frame t we use the same technique to compute the cumulative minimum costs $\mathbf{B}_t$ from the bottom to the top row. Then, the complete vertical seam costs for frame t are given by



(a) partial cumulative seam cost



(b) total cumulative seam cost

Figure 2.5: (a) The partial cumulative seam $\mathbf{M}$ cost leads to larger values at the bottom of the frame. (b) The total cumulative seam cost $\hat{\mathbf{M}}$ assigns each pixel the total cost of its minimal vertical seam. Blue indicates low values, Red high values.

$$\hat{\mathbf{M}}|_t = \mathbf{M}|_t + \mathbf{B}_t - \mathbf{C}|_t - \mathbf{M}|_{t-1}(:, \mathbf{I}|_t). \tag{2.12}$$

Note that by adding the result of the top-to-bottom and bottom-to-top result, the cost value of each element as well as the optimal neighbor in the frame before is counted twice. This is corrected by explicitly subtracting these values. The difference is illustrated in Fig. 2.5.

2. The definition of the cumulative minimal cost of a seam in Eq. 2.11 leads to values of $\mathbf{M}$ whose magnitude increases linear wrt. the frame number. This imbalance gives

preference to seams at the start of the video. As a result, after processing a frame t, we normalize $\hat{\mathbf{M}}|_t$ by $2m$, that is the amount of elements summed per vertical seam.

After computing the cost measure $\hat{\mathbf{M}}$ for every frame, we can use the constructed matrix $\mathbf{I}$ to determine an approximate surface. For this reason we show that $\mathbf{I}$ has a very important property. If $\mathbf{s} = (p_1, p_2, ..., p_m)^T,\ p_i \in \mathbb{N}$ denotes the x-offsets of a minimal vertical seam in frame t,

$$\mathbf{I}(\mathbf{s}) = (\mathbf{I}(p_1, 1, t), \mathbf{I}(p_2, 2, t), ..., \mathbf{I}(p_m, m, t)^T) \tag{2.13}$$

denotes a vertical seam in frame $t-1$ that is a neighbor of $\mathbf{s}$, i.e. satisfies Eq. 2.3. However, $\mathbf{I}(\mathbf{s})$ is not necessarily a minimal vertical seam in frame $t-1$, although in practice it will be close to one. Therefore $\mathbf{I}(\mathbf{I}(\mathbf{s}))$ is not generally a seam, i.e. it violates the smoothness condition of Eq. 2.3 along the y-dimension.

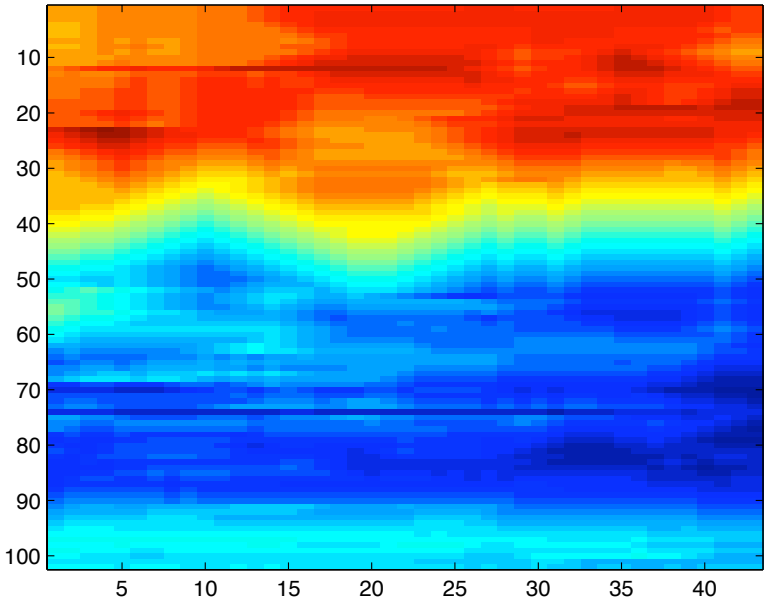
Let $\mathbf{s_r}$ be the minimum vertical seam in frame r that is obtained by backtracking $\mathbf{M}$, then we can approximate the optimal surface $\mathbf{S}^*$ by

$$\hat{\mathbf{S}} = [\mathbf{I}^{r-1}(\mathbf{s_r}) \mid \mathbf{I}^{r-2}(\mathbf{s_r}) \mid ... \mid \mathbf{I}(\mathbf{s_r}) \mid \mathbf{s_r}] \tag{2.14}$$

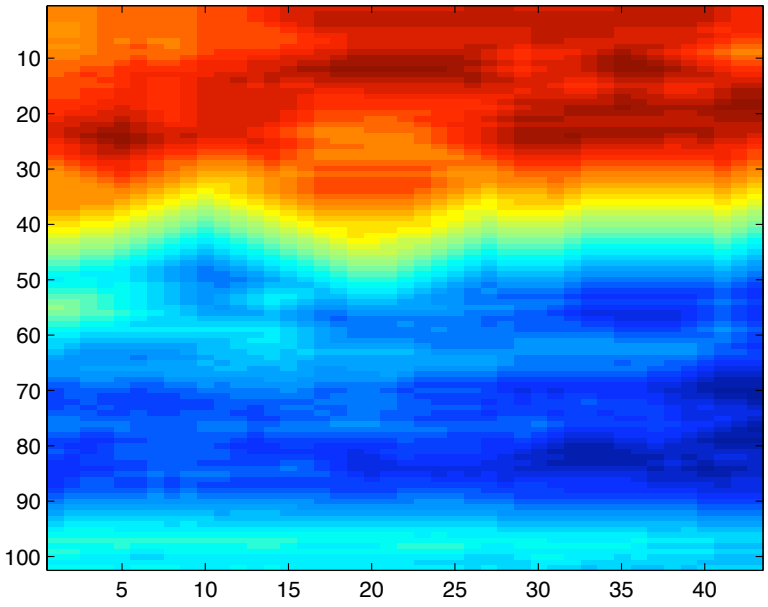
As stated above, $\hat{\mathbf{S}}$ will have some discontinuities along the y-dimension of the video. We describe how to fix these in the next section.

2.4 Smoothing the approximate surface

A height profile of a typical approximate surface computed by the approach described above is shown on top of Fig. 2.6. A possible solution to smooth the discontinuities is replacing them by the average of the involved neighbors, which is similar to convolution with a Gaussian. However, this might result in surfaces that cut through a volume of high costs. Therefore, we use a weighted average depending on our cost measure $\mathbf{C}$. Each point $\mathbf{p}$ on the surface is assigned a weight that is the inverse of its corresponding cost $\mathbf{C}(\mathbf{p})$. The weights are thresholded to exclude very small and large values and scaled linearly to lie in the range of 1 to 20.



(a) Surface with discontinuities



(b) Surface after smoothing

Figure 2.6: (a) Approximate optimal surface with discontinuities (e.g. see row 12 and 75 in (a)). Red denotes high x-values, blue low x-values. (b) Final surface after 23 weighted smoothing iterations.

The technique is successively applied to $\hat{\mathbf{S}}$ until it satisfies Eq. 2.3. Generally, 20-50 iterations are needed to obtain a surface that satisfies the smoothness constraint in Eq. 2.3. A typical result is shown at the bottom of Fig. 2.6. We will now describe how to use $\hat{\mathbf{S}}$ to obtain a representation that allows content-aware resizing of video in real-time.

2.5 Building multi-view videos

The property, that successively computed surfaces are disjunct allows content-aware real-time resizing of videos, very similar to the real-time resizing of images shown by Avidan and Shamir [3].

Let $\mathbf{S_l}$ denote the surface for resizing the video from width l to $l-1$. Furthermore, we will consider a matrix $\mathbf{R} \in \mathbb{N}^{m \times n \times r}$ of identical size as the video volume $\mathbf{V} \in \mathbb{R}^{m \times n \times r}$ with all values initialized to zero. Our goal is to save the information of all carving surfaces in $\mathbf{R}$. Hence, for each pixel $\mathbf{p} \in \mathbf{S_l}$ we set the corresponding entries in $\mathbf{R}$ to l. Then, a multi-view video is defined as the pair $(\mathbf{V}, \mathbf{R})$.

Each entry in $\mathbf{R}$ determines the target size at which the corresponding pixel would be removed by some surface. As a result, if we want to resize the video to a target width of k, we simply determine which entries in $\mathbf{R}$ are smaller or equal to k and copy only the corresponding pixels from $\mathbf{V}$ to the desired target frame. Therefore, resizing a video only involves comparing the entries of $\mathbf{R}$ to a constant and copying the corresponding pixels. These operations can easily be performed in real-time even by low-end processors e.g. used in mobile devices.

In general we focus on altering the size of a video within 50% of its original dimensions. Therefore, the matrix $\mathbf{R}$ is sparse and can be compressed by common techniques like run-length encoding (RLE) or 3D spatial hashing [12].

Currently, our approach of building multi-view videos only works for altering either the width or the height of the video. The main challenge with constructing a representation that works in both dimensions originates from the following fact. All horizontal surfaces that we compute for a $m \times n \times r$ video volume are of size $n \times r$. Respectively, all vertical surfaces are of size $m \times r$. Removing a horizontal surface from the video changes its size to

$(m-1) \times n \times r$. If we want to remove one of the previously computed vertical surfaces we face the problem of how to reduce the $m \times r$ surface to $(m-1) \times r$. Assume the intersection of each horizontal surface with each vertical surface forms a temporal seam, i.e. a seam that has exactly one pixel in each frame. Then, we can simply remove the common pixel in each frame to reduce the $m \times r$ surface to dimension $(m-1) \times r$. However, this is usually not the case and while constraining all surfaces to fulfill this constraint is feasible it greatly limits their degrees of freedom which results in more spatial and temporal artifacts.

It can be actually questioned whether real-time content-aware resizing in both dimensions is required. We can scale a video to every desired target size by using uniform scaling to match one of its dimension to the desired target size and using content-aware resizing to match the other.

2.6 Breaking surfaces apart

As we will show in detail in our experimental section, objects that are moving rather fast pose a problem to our current approach as well as to every other approach that relies on carving of smooth surfaces. The reason behind this is that the smooth surface can only move by maximally one pixel between one frame to the adjacent one. In case the objects exhibit a larger motion than one pixel we are likely to compute a surface that cuts through the moving objects.

In the extreme case of an object moving from the left to the right border in a video, every surface will cut through the space-time volume of the moving object. In order to overcome the resulting distortions there is no other solution than to break the surface along its temporal axis while keeping the discontinuities per frame minimal.

We propose a cascading approach to solve this problem as demonstrated in Fig. 2.7. Each surface is constrained to be only k frames long, where k depends on the magnitude of motion within the video. Fast paced camera motions require a k between 3 to 5 frames, while for slower motions we select a k between 10-30 frames. In case of static background $k = \infty$ gives the best results, because every single temporal discontinuity is noticeable.

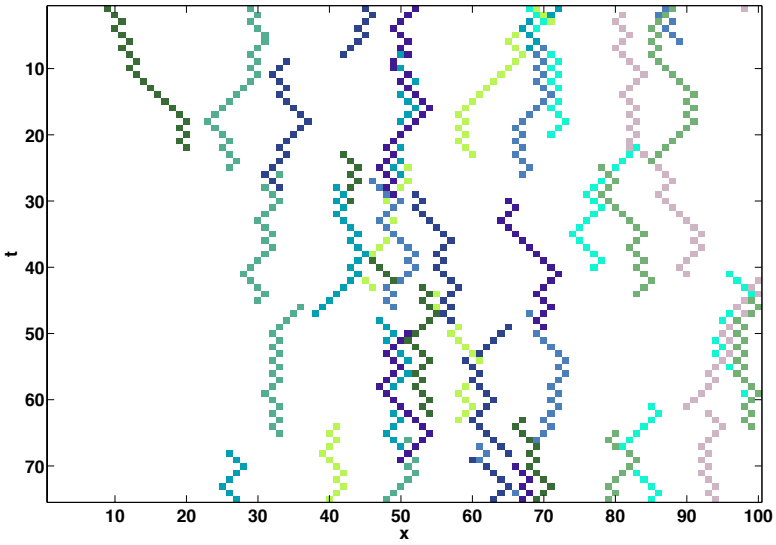


Figure 2.7: Cascading surfaces (shown for a fixed y). Surfaces belonging to same set are given the same color.

We compute surfaces of length k every k-th frame. To minimize the introduced discontinuities per frame after each iteration we shift the starting frame of each surface by one. Consequently, the at the $i-th$ iteration the first surface will only be $(k-i) \mod k$ frames long and the second surface will start at frame number $i \mod k$. We will examine the effect of our cascading surfaces approach in the experimental section in chapter 4.

2.7 Enlarging videos and different cost measures

To enlarge a video we can reuse our representation of multi-view videos described above. Assume we want to increase the width of a video with spatial dimensions $n \times m$ to $k > n$. We use the matrix $\mathbf{R}$ and determine the entries of $\mathbf{R}$ that are greater than $-k+n$ and simply duplicate those pixels. Note, that repetitively duplicating the same surface of minimal cost would produce very noticeable band-like artifacts in the resulting video. That is why we have to chose $k-n$ different surfaces.

We are not limited to the cost measure in Eq. 2.4. We experimented with different different descriptions of the saliency cost $\mathbf{C}$ and found the following variation to be useful:

$$\mathbf{C} = \frac{|\partial_x \mathbf{V}| + |\partial_y \mathbf{V}|}{g(|\partial_t \mathbf{V}|)} \tag{2.15}$$

where g is a function that weights the magnitude of temporal derivate by 0.1 and thresholds the result to lie in the interval [1,10]. The motivation behind this formulation is that seams can usually be hidden in regions that exhibit significant motion between two frames. We show the usefulness of this measure in an example in our experimental evaluation in chapter 4.

Other variations of saliency might use additional information like the one supplied by face detectors. However, we feel that this only improves results on a small subset of videos. Given the subjective nature of content-aware resizing - what one person considers important does not have to be important for another person - we think that a definite solution to this problem requires additional user input. However, the design of a system that enables the user to conveniently specify what he thinks is important in a video, poses a challenging problem and is beyond the scope of this thesis.

CHAPTER III

CONTENT AWARE VIDEO RESIZING POSED AS A LINEAR MINIMIZATION PROBLEM

3.1 Linear constraints in Video Resizing

In order to compare our iterative greedy approach to global optimization approaches of other authors, we re-implemented Wolf et al.'s [18] system. We will describe their approach and provide the details of our efficient solution by explicitly computing the normal equations. Lastly, we will mention our extensions to improve robustness.

Let us denote a video's frame width by w and its frame height by h. Similar to our approach, every pixel $p = (i, j),\ i = 1 \dots w, j = 1 \dots h$ is associated with a local saliency or energy measure $S(p) = S_{i,j}$ represented as a saliency matrix $\mathbf{S} = (S_{i,j})$. In order to resize the video to its new target frame size $t_w \times t_h$, we have to calculate for each pixel (i, j) in the original frame its new position $(x_{i,j}, y_{i,j})$ in the target frame. This problem can be posed as a linear minimization problem. Because the target locations $x_{i,j}$ and $y_{i,j}$ are independent of each other, we will focus on describing the layout of model matrix that solves for $x_{i,j}$. Solving for $y_{i,j}$ can easily achieved by transposing the saliency matrix $\mathbf{S}$. In the following description, we will denote by $x_j, j = 1 \dots h$ the row-vector $[x_{1,j}, x_{2,j}, \dots, x_{w,j}]^T$

Wolf et al. [18] propose several linear constraints to achieve a content-aware, spatially and temporally coherent resized version of the video. The constraints are as follows.

3.1.1 Border constraint

Pixels at the left and right border of the source frame should be mapped to the left and right border of the target frame. Otherwise, not the whole available space in the target frame might be used. Therefore we have:

$$x_{1,j} = e_1^T x_j = 1 \text{ and } x_{w,j} = e_w^T x_j = t_w,\ \forall j = 1 \dots h \tag{3.1}$$

with $e_1 = (1, 0, \ldots, 0)^T$ and $e_w = (0, \ldots, 0, 1)$ being the first and last vector of the standard basis of $\mathbb{R}^m$. In matrix notation this translates to

$$\begin{pmatrix} e_1^T & & & & & \\ & e_1^T & & & & \\ & & \ddots & & & \\ & & & e_1^T & & \\ & & & & e_1^T & \\ e_w^T & & & & & \\ & e_w^T & & & & \\ & & \ddots & & & \\ & & & e_w^T & & \\ & & & & e_w^T & \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{h-1} \\ x_h \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ t_w \\ t_w \\ \vdots \\ t_w \\ t_w \end{pmatrix}. \tag{3.2}$$

3.1.2 Neighboring constraint

We want to preserve the ordering of pixels within a row to avoid spatial artifacts in the resulting resized frame. That means, that neighboring pixels in the source frame are constrained to be neighboring pixels in the target frame as well. We will weight this constraint by each pixel's saliency, i.e. unimportant pixels will be squished together or pulled apart while salient pixels will have a spacing of exactly one pixel like in the original frame. Therefore we have

$$\begin{aligned} S_{i,j}(x_{i,j} - x_{i-1,j}) &= 1 \cdot S_{i,j} \quad \forall i = 2 \ldots w,\ j = 1 \ldots h \qquad \text{and} \\ S_{i,j}(x_{i,j} - x_{i+1,j}) &= -1 \cdot S_{i,j} \quad \forall i = 1 \ldots w-1,\ j = 1 \ldots h. \end{aligned} \tag{3.3}$$

In matrix notation this translates to

$$\begin{pmatrix} S_{i,j} & \\ & S_{i,j} \end{pmatrix} \cdot \begin{pmatrix} -1 & 1 & \\ & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_{i-1,j} \\ x_{i,j} \\ x_{i+1,j} \end{pmatrix} = \begin{pmatrix} S_{i,j} & \\ & S_{i,j} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{3.4}$$

We can express this relation for a specific row j in the following way:

$$\mathbf{D_j} \cdot \underbrace{\begin{pmatrix} 1 & -1 & & & \\ -1 & 1 & & & \\ & 1 & -1 & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & -1 & 1 & \\ & & & 1 & -1 \\ & & & -1 & 1 \end{pmatrix}}_{:=\mathbf{A}} \cdot \begin{pmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{h-1,j} \\ x_{h,j} \end{pmatrix} = \mathbf{D_j} \cdot \underbrace{\begin{pmatrix} -1 \\ 1 \\ -1 \\ \vdots \\ \vdots \\ -1 \\ 1 \end{pmatrix}}_{:=k} \tag{3.5}$$

with $\mathbf{D_j}$ being the saliency weighting diagonal matrix for row j

$$\mathbf{D_j} = \begin{pmatrix} S_{1,j} & & & & & & & \\ & S_{2,j} & & & & & & \\ & & S_{2,j} & & & & & \\ & & & \ddots & & & & \\ & & & & \ddots & & & \\ & & & & & S_{w-1,h} & & \\ & & & & & & S_{w-1,h} & \\ & & & & & & & S_{w,h} \end{pmatrix}. \tag{3.6}$$

In short, we can describe the neighboring constraint for each row j as:

$$\mathbf{D_j A} x_j = \mathbf{D_j} k \tag{3.7}$$

3.1.3 Column smoothness constraint

While the above described neighboring constraint preserves the ordering of pixels *within* a row it does not model coherence *among* the rows. Without an additional constraint pixels that are close in the original image but in different rows would have the freedom to move far apart. This results in zig-zag or wave-like artifacts. As a consequence, a row smoothness constraint is introduced that maps pixels in the same column of the source frame to the same column in the target frame

$$\alpha \cdot (x_{i,j} - x_{i,j-1}) = 0 \quad \forall i = 1 \dots w, \; j = 2 \dots h, \tag{3.8}$$

where α is an empirically determined weighting parameter with $\alpha \in [0, 1]$. For two adjacent rows x_{j-1} and x_j this constraint can be rewritten as

$$\begin{pmatrix} \alpha \mathbf{I}_w & -\alpha \mathbf{I}_w \end{pmatrix} \cdot \begin{pmatrix} x_{j-1} \\ x_j \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.9}$$

where $\mathbf{I}_w$ denotes the w-dimensional identity matrix in $\mathbb{R}^m$.

The same constraint is imposed between the first and the last row of the frame to prevent drifting. A different weight $\gamma \in [0, 1]$ is used instead of α

$$\begin{pmatrix} \gamma \mathbf{I}_w & -\gamma \mathbf{I}_w \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_h \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.10}$$

3.1.4 Temporal smoothness constraint

A solution to prevent noticeable temporal artifacts among frames is to make the warping for the current frame similar to the warping of the previous frame. Specifically, if $(\hat{x}_{i,j}, \hat{y}_{i,j})$ denotes the solution of the previous frame, the current solution $(x_{i,j}, y_{i,,j})$ should satisfy

$$\beta(x_{i,j} - \hat{x}_{i,j}) = 0 \quad \forall i = 1 \dots w, \; j = 1 \dots h, \tag{3.11}$$

where β is again an empirically determined parameter. In matrix notation that is

$$\beta \mathbf{I}_w \cdot x_j = \beta \hat{x}_j. \tag{3.12}$$

3.2 Combining the constraints into one linear system

To solve for the x-target position $x_{i,j}$ for every pixel (i, j) in the source frame, we can satisfy the border constraint, the neighboring constraint, the column smoothness constraint and the temporal constraint by combing their corresponding linear equations into one large system:

$$
\begin{pmatrix}
e_1^T & & & & & \\
& e_1^T & & & & \\
& & \ddots & & & \\
& & & e_1^T & & \\
& & & & e_1^T & \\
e_w^T & & & & & \\
& e_w^T & & & & \\
& & \ddots & & & \\
& & & e_w^T & & \\
& & & & e_w^T & \\
\mathbf{D_1 A} & & & & & \\
& \mathbf{D_2 A} & & & & \\
& & \ddots & & & \\
& & & \mathbf{D_{h-1} A} & & \\
& & & & \mathbf{D_h A} & \\
\alpha \mathbf{I}_w & -\alpha \mathbf{I}_w & & & & \\
& \alpha \mathbf{I}_w & -\alpha \mathbf{I}_w & & & \\
& & \ddots & \ddots & & \\
& & & \alpha \mathbf{I}_w & -\alpha \mathbf{I}_w & \\
\beta \mathbf{I_w} & & & & & \\
& \beta \mathbf{I_w} & & & & \\
& & \ddots & & & \\
& & & \beta \mathbf{I_w} & & \\
& & & & \beta \mathbf{I_w} & \\
\gamma \mathbf{I_w} & & & & -\gamma \mathbf{I_w} &
\end{pmatrix}
\cdot
\begin{pmatrix}
x_1 \\ x_2 \\ \vdots \\ x_{h-1} \\ x_h
\end{pmatrix}
=
\begin{pmatrix}
1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ t_w \\ t_w \\ \vdots \\ t_w \\ t_w \\ \mathbf{D_1} k \\ \mathbf{D_2} k \\ \vdots \\ \mathbf{D_{h-1}} k \\ \mathbf{D_h} k \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \beta \hat{x}_1 \\ \beta \hat{x}_2 \\ \vdots \\ \beta \hat{x}_{h-1} \\ \beta \hat{x}_h \\ 0
\end{pmatrix}
\tag{3.13}
$$

or in short

$$\mathbf{C}x = b. \tag{3.14}$$

If we denote by $n := w \cdot h$ the number of total pixels within a frame, then $\mathbf{C}$ is approximately of dimension $4n \times n$.

3.3 Solving $\mathbf{C}x = b$

We found empirically that the condition number of $\mathbf{C}$ is $O(10^3)$, if α, β and γ are chosen to be of similar magnitude ~ 0.5. This is mainly due to the block matrices $\mathbf{D_j A}$, the other submatrices of $\mathbf{C}$ hold values of magnitude around 1. $\mathbf{D_j}$ is a diagonal matrix which entries are the saliency values $S_{i,j}$. In our case, the saliency is defined as the sum of the absolute values of the image gradients and therefore the ratio of the largest to the smallest entry is ~ 500,[1] which given the structure of $\mathbf{A}$ in Eq. 3.5 explains the observed condition number. While there are many ways to solve a linear minimization problem, we will determine the solution of Eq. 3.14 by solving the normal equation $\mathbf{C}^T\mathbf{C}x = \mathbf{C}^T b$. While being numerical inferior to approaches like the QR-decomposition, in our case we can calculate $\mathbf{C}^T\mathbf{C}$ explicitly, therefore reducing the amount of storage by a factor of at least 4 and saving computation time. Note, that the condition number of the normal equation is $O(10^6)$. Hence we can obtain a sub-pixel accurate solution if we carry out all computations in double precision arithmetic, although we found that single precision is sufficient for most practical cases.

3.3.1 Computing $\mathbf{C}^T\mathbf{C}$

We can calculate $\mathbf{C}^T\mathbf{C}$ by using the block structure of $\mathbf{C}$ and obtain a block matrix of the following form

[1] We threshold the saliency values to be at least 1

$$\mathbf{C}^T\mathbf{C} = \begin{pmatrix} \mathbf{U_1} & \alpha^2\mathbf{I_w} & & & \gamma^2\mathbf{I_w} \\ \alpha^2\mathbf{I_w} & \mathbf{U_2} & \alpha^2\mathbf{I_w} & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha^2\mathbf{I_w} & \mathbf{U_{h-1}} & \alpha^2\mathbf{I_w} \\ \gamma^2\mathbf{I_w} & & & \alpha^2\mathbf{I_w} & \mathbf{U_h} \end{pmatrix} \tag{3.15}$$

with $\mathbf{U_i}$ defined as

$$\mathbf{U_i} = e_1e_1^T + e_we_w^T + \mathbf{A}^T{\mathbf{D_i}}^2\mathbf{A} + \beta^2\mathbf{I_w} + \begin{cases} (\alpha^2+\gamma^2)\mathbf{I_w} & \text{if } i=1 \text{ or } \ i=h \\ 2\alpha^2\mathbf{I_w} & \text{otherwise} \end{cases}. \tag{3.16}$$

$\mathbf{D_j}^2$ is a diagonal matrix with elements $\sigma_{1,j}, \sigma_{2,j}, \sigma_{2,j}, \ldots, \sigma_{w-1,j}, \sigma_{w-1,j}, \sigma_{w,j}$ where $\sigma_{i,j} = S_{i,j}^2$. It can be shown, that the expression $\mathbf{A}^T{\mathbf{D_i}}^2\mathbf{A}$ is identical to the matrix

$$\begin{pmatrix} \sigma_{1,j}+\sigma_{2,j} & -(\sigma_{1,j}+\sigma_{2,j}) & & & \\ -(\sigma_{1,j}+\sigma_{2,j}) & \sigma_{1,j}+2\sigma_{2,j}+\sigma_{3,j} & \ddots & & \\ & -(\sigma_{2,j}+\sigma_{3,j}) & \ddots & & \\ & & \ddots & \sigma_{h-2,j}+2\sigma_{h-1,j}+\sigma_{h,j} & -(\sigma_{h-1,j}+\sigma_{h,j}) \\ & & & -(\sigma_{h-1,j}+\sigma_{h,j}) & \sigma_{h-1,j}+\sigma_{h,j} \end{pmatrix}. \tag{3.17}$$

3.3.2 Computing $\mathbf{C}^T b$

By again taking advantage of the block structure of $\mathbf{C}$ we can show, that $\mathbf{C}^T b$ is identical to the following expression:

$$\mathbf{C}^T b = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_{h-1} \\ \hat{b}_h \end{pmatrix} \tag{3.18}$$

with

$$\hat{b}_j = e_1 + t_w \cdot e_w + \mathbf{A}^T{\mathbf{D_j}}^2 k + \beta^2\hat{x}_j \tag{3.19}$$

where $\mathbf{D_j}^2$ is defined as in the previous paragraph. Computing $\mathbf{A}^T\mathbf{D_j}^2 k$ yields to

$$\begin{pmatrix} -(\sigma_{1,j} + \sigma_{2,j}) \\ \sigma_{1,j} - \sigma_{3,j} \\ \sigma_{2,j} - \sigma_{4,j} \\ \vdots \\ \sigma_{h-2,i} - \sigma_{h,i} \\ \sigma_{h-1} + \sigma_h \end{pmatrix} \tag{3.20}$$

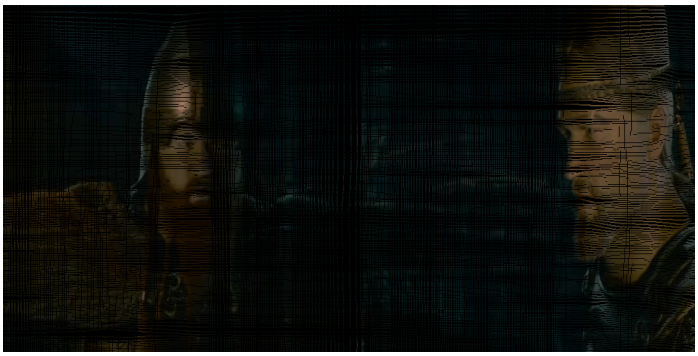
3.4 Extensions and solution details

Having computed $\mathbf{C}^T\mathbf{C}$ and $\mathbf{C}^T b$ we can use any sparse matrix solver to solve for the target position $x_{i,j}$. Because $\mathbf{C}^T\mathbf{C}$ is symmetric and positive definite we can use sparse Cholesky decomposition as well as LU decomposition. We experimented with the linear solvers packages CHOLMOD Version 1.7 [5], which implements supernodal sparse Cholesky factorization and was shown to be one of the fastest sparse Cholesky factorizers currently available [9]. We also applied SuperLU Version 3.0 [7] to our system, which computes a sparse LU decomposition. Similar to the difference in computational complexity[2], we found that CHOLMOD solved the system by a factor of 2 faster than SuperLU with an average time of $\sim 1s$ per frame, if applied to a widescreen video trailer with a frame size of approximately 600×250 pixels.

After we solved for the new target position x, we use forward warping with blending to obtain the resized target frame. In case we use Wolf et al.'s [18] technique to enlarge a video frame the forward warping can lead to holes. Rather than restating the constraints in terms of backward warping we can use an in-painting technique to fill the holes. Generally, the holes are not larger than 1-2 pixel. Consequently, instead of using an established non-texture in-painting technique [4, 17] we simply average the values of known neighbors for each pixel in a hole. To robustly deal with larger holes which occasionally occur, we represent the holes in a frame as a binary mask B and successively shrink B to $B \leftarrow \text{erosion}(B)$. Only the pixels removed in each step $R \leftarrow B \setminus \text{erosion}(B)$ are in-painted, which is similar to

[2] $\frac{1}{3}m^3$ for Cholesky vs. $\frac{2}{3}m^3$ LU decomposition in case of dense systems

diffusion. An example of our technique is shown in Fig. 3.1.



(a) Forward warping in case of enlargement (both dimensions)

(b) After inpainting.

Figure 3.1: Example of our inpainting technique. Details given in the text.

As we show in our experimental evaluation in chapter 4 Wolf et al.'s approach suffers from what we call *the collapse of the solution*. It means that a set of pixels gets mapped to the same location in the target frame. This produces good-looking results in case the collapse happens at the border of the video frame, which is equivalent to cropping unsalient regions. However, in case the collapse happens in the middle of the video frame it results in unpleasant artifacts.

We believe that the main problem lies in the definition of the neighboring constraint in Eq. 3.3, specifically that only the saliency is used to weight the constraint. As a result the neighboring constraint is enforced in salient regions, while it is neglected in non-salient ones. Instead of neglecting the neighboring constraint in non-salient regions and therefore giving the solution the freedom to collapse, we want the non-salient region to be squished together

or stretched out depending on whether we shrink or enlarge the video frame. We still weight each neighboring constraint by the saliency of the corresponding pixel but substitute the 1 on the right hand side of Eq. 3.3 by a saliency-dependent measure. We experimented with various measures and found that a convex combination of a guaranteed base distance and a saliency dependent sigmoid function performes best. In case of shrinking a video we use a base distance 0.5 pixels and a convex weight of 0.3 for the base distance. A convex weight of 0 is identical to Wolf et al.'s original approach while a convex weight larger than 0.5 usually leads to non-uniform rescaling. An example of the collapsing solution as well as a comparision to our technique is shown in Fig. 3.2.

(a) Collapse of the solution in Wolf et al.'s approach

(b) Our approach

Figure 3.2: Original frame is resized to 70%. The collapse of the solution results in the interleaving of two regions, recognizable as black stripes on right side of the frame. Our solution does not suffer from the same artifacts.

CHAPTER IV

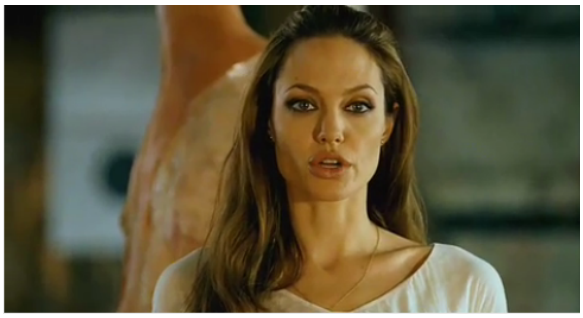
EXPERIMENTS AND RESULTS

We ran many experiments to test our approach on a broad variety of videos. An empirical evaluation is difficult because there is no metric to evaluate how good or bad a content-aware resizing technique performs. Therefore, we will explain characteristics of our technique and discuss success and failure cases on the basis of examples.

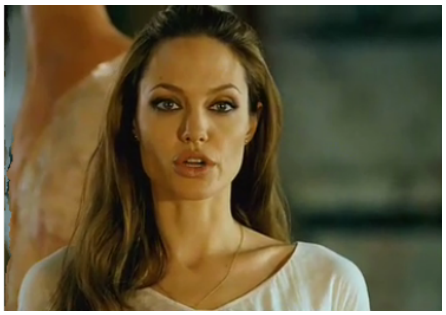
All examples in this chapter are prepared with identical parameter settings as described in the chapters before. We did not tune our parameters for an individual video, if we changed a specific part of our algorithm such as the saliency measure we document it and show the difference to our original implementation. As noted earlier, our multi-view video representation allows us to resize videos in real-time. Computation of the multi-view videos is fast - a representation that allows changing the width of a 60 frame long 450×200 video to a width of 70% to 130% of its original width can be computed in around a minute on a common 2.4 GHz laptop.

We start our discussion by evaluating how well our technique shrinks homogenous regions. Both source frames in Fig. 4.1 contain large homogenous regions that are successfully squished together, while the highly textured object of interest is preserved.

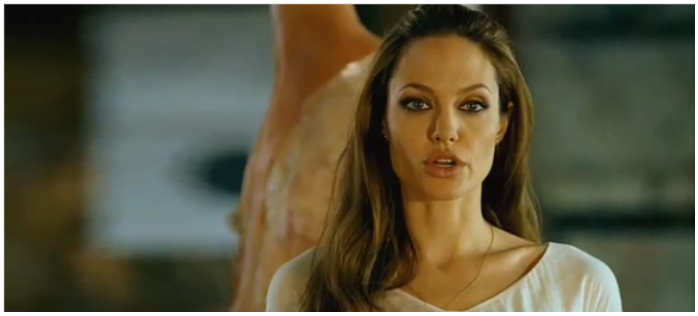
In Fig. 4.2 we compare our approach to our implementation of Wolf et. al's. [18] optimization approach. We use the same saliency measure as input for both techniques. In this example most regions are highly textured and we see that Wolf et al.'s approach is more similar to non-uniform rescaling than content-aware rescaling. However, in videos where a smart cropping technique can be applied, Wolf et al.'s approach performs better than ours as shown in Fig. 4.3. It is interesting to note that Wolf et al.'s approach achieves temporal coherence by constraining the solution for the current frame only on the result of the last frame, whereas we use the whole video volume. We will investigate this in future work.



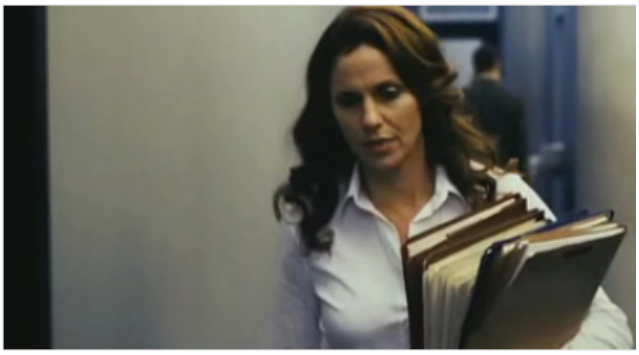
(a) Original frame



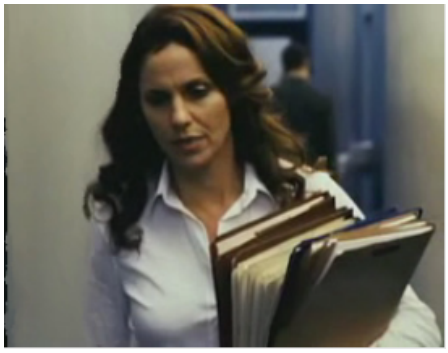
(b) Frame resized to 75%.



(c) Frame resized to 120% of original width



(d) Original frame



(e) Frame resized to 70%.

(f) Frame resized to 130% of original width

Figure 4.1: Correctness test of our approach. Top example: The low textured region on the right hand side is carved away. Original frame taken from movie *Wanted* - Copyright 2008 Universal Pictures. Bottom example: The homogenous region on the right hand side is carved away. Original frame taken from the movie *88 minutes* - Copyright 2008 by TriStar Pictures.

(a) First frame

(b) Frame 69

(c) First frame resized to 70% by our approach

(d) Frame 69 resized to 70% by our approach

(e) First frame resized to 70% using Wolf et al.'s approach

(f) Frame 69 resized to 70% using Wolf et al.'s approach

Figure 4.2: Comparison between our method (2nd row) and Wolf et al.'s [18] approach. We used the same saliency as input for both methods. Original frames taken from the movie *Beowulf* - Copyright 2007 by Paramount Pictures.

(a) Original frame

(b) Frame frame resized to 70% using Wolf et al.'s approach

(c) Frame resized to 70% by our approach

Figure 4.3: Comparison between our method (right) and Wolf et al.'s [18] approach (left). We used the same saliency as input for both methods. Original frames taken from the movie *Sweeney Todd: The Demon Barber of Fleet Street* - Copyright 2007 by DreamWorks Pictures.

As mentioned before, a general problem with both techniques is that they produce a smooth solution, while a video is only piecewise smooth with discontinuities at occlusion boundaries. Therefore, in case objects move significantly the smooth solution is not able to keep up with their motion and intersects with the spatio-temporal volume defined by the moving object. An example of this behavior is shown in Fig. 4.4.

In chapter 2 we proposed a method of breaking surfaces along the temporal dimension to resolve this problem. Fig. 4.5 shows two fast moving planes on a homogenous background. Both planes move from the left in frame 1, to the right in frame 16 and end in the middle of frame 32. Our cascading surface technique is able to carve around the spatio-temporal volumes defined by the planes' motion without introducing spatial artifacts. On the contrary, using only continuous surfaces or Wolf et al.'s approach results in distortions as shown in Fig. 4.6.

However, our cascading surface technique is not perfect. In Fig. 4.7 we show an example that while being free of spatial distortions has temporal incoherency. Note, that is a very challenging example. A possible solution would have to alter the spacing of the letters dynamically to resize the whole content to the desired target size.

A problem that still needs to be solved is a adequate definition of saliency. However, it can be doubted that this problem can be overcome by more sophisticated definition of saliency, like learning approaches or taking advantage of face detectors. Without additional user input, it seems impossible to produce results that are satisfying in all cases, as we show in Fig. 4.8. In this example, the background is highly textured while the person's coat is mostly of solid black color. Any approach that is based on image gradients will fail here. A possible solution would be to supply the user with an interface to conveniently specify that the person is the object of interest.

In Fig. 4.9 we applied the variation of our definition of saliency as described by Eq. 2.15. The results are significantly better than those achieved by different approaches.

We finish our evaluation by showing more successful applications of our content-aware resizing technique in Fig. 4.10, 4.11 and 4.12.

(a) First frame

(b) First frame resized to 70% of the original width

(c) Frame 40

(d) Frame 21 resized to 70% of the original width

(e) First frame resized to 70% by using Wolf et al.'s [18] approach

(f) Frame 40 resized to 70% by using Wolf et al.'s [18] approach

Figure 4.4: Failure case. The motion between frame 1 and 40 is large, therefore every carved surface will distort the resulting frame. Similar artifacts are produced by the global optimization method of Wolf et al. [18]. Original frame taken from the movie *88 minutes* - Copyright 2008 by TriStar Pictures.

(a) Original frame 1

(b) Frame 1 rescaled to 70% by cascading surface approach

(c) Original frame 16

(d) Frame 16 rescaled to 70% by cascading surface approach

(e) Original frame 32

(f) Frame 32 rescaled to 70% by cascading surface approach

Figure 4.5: Content-aware resizing by using cascading surfaces with a length of $k = 3$ frames. Note, that every continous surface would intersect the space-time volume defined by the motion of the two planes. Original frame taken from the movie *Valkyrie* - Copyright 2008 by Metro-Goldwyn-Mayer.

(a) Frame 1 rescaled to 70% by continuous surface approach

(b) Frame 16 rescaled to 70% by continuous surface approach

(c) Frame 32 rescaled to 70% by continuous surface approach

(d) Frame 1 rescaled to 70% by Wolf et al.'s approach

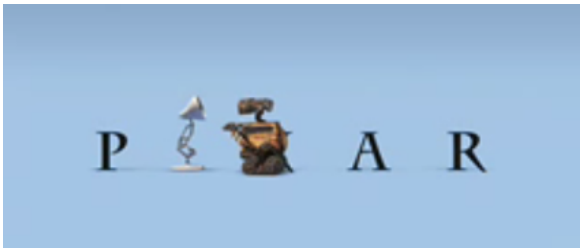
(e) Frame 16 rescaled to 70% by Wolf et al.'s approach

(f) Frame 32 rescaled to 70% by Wolf et al.'s approach

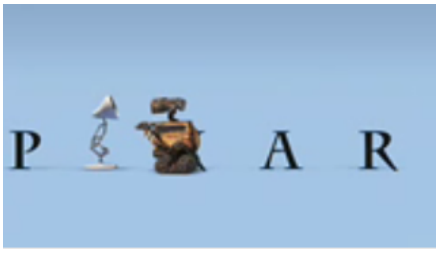
Figure 4.6: Comparison to Fig.4.5. Top row: Content aware resizing by using continuous surfaces. Bottom row: Wolf et al.'s [18] approach. Both approaches do not yield satisfactory results. Original frame taken from the movie *Valkyrie* - Copyright 2008 by Metro-Goldwyn-Mayer.

(a) First frame

(b) First frame resized to 75% of the original width



(c) Frame 40



(d) Frame 40 resized to 75% of the original width

Figure 4.7: Failure case for cascading surfaces. While no spatial artifacts are introduced temporal coherence is lost. Notice the movement of the P between frame 1 and 40 - the result is similar to pan and scan. Original frame taken from the movie *WALL-E* - Copyright 2008 by Pixar Animation Studios.

(a) Original frame

(b) Frame resized to 70% by using our surface carving approach

(c) Frame resized to 70% by using Wolf et al.'s approach

Figure 4.8: Failure case caused by a gradient based definition of local saliency. All gradient based approaches fail because the object of interest is low textured while the less important background is highly textured. Without additional user input we might not be able to solve this problem. Original frame taken from movie *No Country for Old Men* - Copyright 2007 Miramax Films.

(a) First frame

(b) First frame resized to 70% by using the saliency measure in Eq. 2.15

(c) Frame 15

(d) Frame 21 resized to 70% by using the saliency measure in Eq. 2.15

(e) First frame resized to 70% by using original saliency measure in Eq. 2.4

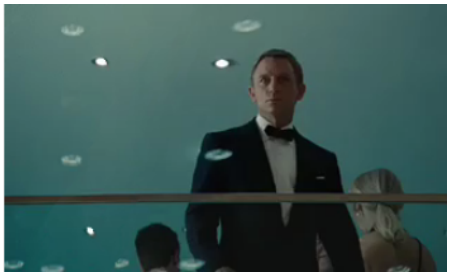
(f) Frame 15 resized to 70% by using original saliency measure in Eq. 2.4

(g) First frame resized to 70% by using Wolf et al.'s approach

(h) Frame 15 resized to 70% by using Wolf et al.'s approach

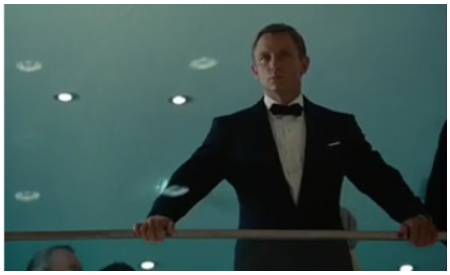
Figure 4.9: Comparison between different saliency measures. Notice the large motion of the background between the two frames - the camera pans around the actress keeping her centered within each frame. Using the variation of our saliency measure (Eq. 2.15) achieves better results. Original frame taken from the movie *The Duchess* - Copyright 2008 by Paramount Vantage.

(a) First frame



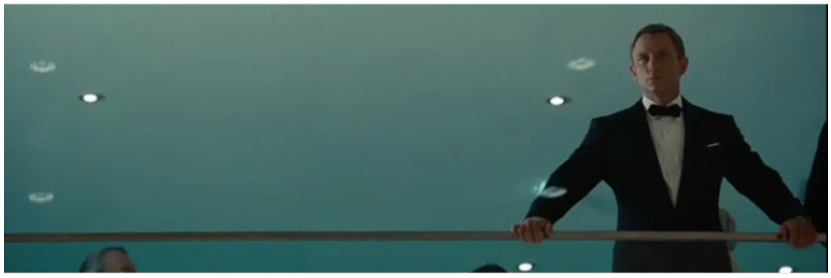
(b) First frame resized to 70% of the original width

(c) Frame 38



(d) Frame 38 resized to 70% of the original width

(e) First frame resized to 130% of the original width



(f) Frame 38 resized to 130% of the original width

Figure 4.10: Application of our resizing technique. Original frames taken from the movie *Quantum of Solace* - Copyright 2008 by Metro-Goldwyn-Mayer.

(a) Original frame

(b) Frame resized to 68% of the original width

(c) Frame resized to 144% of the original width

(d) Original frame

(e) Frame resized to 70% of the original width

(f) Frame resized to 130% of the original width

Figure 4.11: Application of our resizing technique. Original frames taken from the movie *Quantum of Solace* - Copyright 2008 by Metro-Goldwyn-Mayer.

(a) First frame

(b) First frame resized to 80% of the original width

(c) Frame 15

(d) Frame 15 resized to 80% of the original width

(e) First frame resized to 120% of the original width

(f) Frame 15 resized to 120% of the original width

Figure 4.12: Application of our resizing technique. Original frames taken from the movie *Wanted* - Copyright 2008 by Universal Pictures.

REFERENCES

[1] Agarwala, A., Zheng, K. C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D. H., and Szeliski, R., "Panoramic video textures," *ACM Transactions on Graphics, SIGGRAPH 2005*, vol. 24, pp. 821–827, Aug. 2005.

[2] Ahmad, M. and Choi, T.-S., "Fast and accurate 3d shape from focus using dynamic programming optimization technique," *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.(ICASSP '05)*, vol. 2, pp. 969–972, 2005.

[3] Avidan, S. and Shamir, A., "Seam carving for content-aware image resizing," in *ACM Transactions on Graphics, SIGGRAPH 2007*, (New York, NY, USA), ACM Press, 2007.

[4] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C., "Image inpainting," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 417–424, ACM Press/Addison-Wesley Publishing Co., 2000.

[5] Chen, Y., Davis, T. A., and Hager, W. W., "Algorithm 8xx: Cholmod, supernodal sparse cholesky factorization and update/downdate," *ACM Trans. Math. Software*, 2006.

[6] Chenjun Tao, Jiaya Jia, H. S., "Active window oriented dynamic video retargeting," in *Workshop On Dymanical Vision ICCV 2007*, 2007.

[7] Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., and Liu, J. W. H., "A supernodal approach to sparse partial pivoting," *SIAM J. Matrix Analysis and Applications*, vol. 20, no. 3, pp. 720–755, 1999.

[8] Fan, X., Xie, X., Zhou, H.-Q., and Ma, W.-Y., "Looking into video frames on small displays," in *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, (New York, NY, USA), pp. 247–250, ACM, 2003.

[9] Gould, N. I. M., Scott, J. A., and Hu, Y., "A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations," *ACM Trans. Math. Softw.*, vol. 33, no. 2, p. 10, 2007.

[10] Kwatra, V., Essa, I., Bobick, A., and Kwatra, N., "Texture optimization for example-based synthesis," *ACM Transactions on Graphics, SIGGRAPH 2005*, vol. 24, no. 3, pp. 795–802, 2005.

[11] Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A., "Graphcut textures: Image and video synthesis using graph cuts," *ACM Transactions on Graphics, SIGGRAPH 2003*, vol. 22, pp. 277–286, July 2003.

[12] Lefebvre, S. and Hoppe, H., "Perfect spatial hashing," *ACM Transactions on Graphics, SIGGRAPH 2006*, vol. 25, pp. 579–588, July 2006.

[13] Liu, F. and Gleicher, M., "Video retargeting: automating pan and scan," in *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, (New York, NY, USA), pp. 241–250, ACM, 2006.

[14] Ran Gal, O. S. and Cohen-Or, D., "Feature-aware texturing," in *Proceedings of Eurographics Symposium on Rendering*, pp. 297–303, 2006.

[15] Rubinstein, M., Shamir, A., and Avidan, S., "Improved seam carving for video retargeting," *ACM Transaction on Graphics, SIGGRAPH 2008*, vol. 27, no. 3, pp. 1–9, 2008.

[16] Setlur, V., Takagi, S., Raskara, R., Gleicher, M., and Gooch, B., "Automatic image retargeting," in *MUM '05: Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, (New York, NY, USA), pp. 59–68, ACM, 2005.

[17] Telea, A., "An image inpainting technique based on the fast marching method," *journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[18] Wolf, L., Guttmann, M., and Cohen-Or, D., "Non-homogeneous content-driven video-retargeting," in *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*, 2007.

[19] Yonatan Wexler, E. S. and Irani, M., "Space-time completion of video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, 2007.